(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 051 420 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.08.2016 Bulletin 2016/31

(21) Application number: **16159584.8**

(22) Date of filing: **08.06.2012**

(51) Int Cl.:
G06F 11/14 *(2006.01)* ***G06F 11/20*** *(2006.01)*
H04L 12/24 *(2006.01)*

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **30.06.2011 US 201113174271**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
12804233.0 / 2 727 287

(71) Applicant: **Microsoft Technology Licensing, LLC**
Redmond, WA 98052 (US)

(72) Inventors:
- **George, Mathew**
  Redmond, WA Washington 98052 (US)
- **Kruse, David M.**
  Redmond, WA Washington 98052 (US)
- **Pinkerton, James T.**
  Redmond, WA Washington 98052 (US)
- **Battepati, Roopesh C.**
  Redmond, WA Washington 98052 (US)
- **Jolly, Tom**
  Redmond, WA Washington 98052 (US)
- **Swan, Paul R.**
  Redmond, WA Washington 98052 (US)
- **Shang, Mingdong**
  Redmond, WA Washington 98052 (US)
- **Lovinger, Daniel Edward**
  Redmond, WA Washington 98052 (US)

(74) Representative: **Grünecker Patent- und Rechtsanwälte**
PartG mbB
Leopoldstraße 4
80802 München (DE)

Remarks:
This application was filed on 10-03-2016 as a divisional application to the application mentioned under INID code 62.

(54) TRANSPARENT FAILOVER

(57) Described are embodiments directed at persistent handles that are used to retain state across network failures and server failovers. Persistent handles are requested by a client after a session has been established with a file server. The request for the persistent handle includes a handle identifier generated by the client. The server uses the handle identifier to associate with state information. When there is a network failure or a server failover, and a reconnection to the client, the handle identifier is used to identify replayed requests that if replayed would create an inconsistent state on the server. The replayed requests are then appropriately handled.

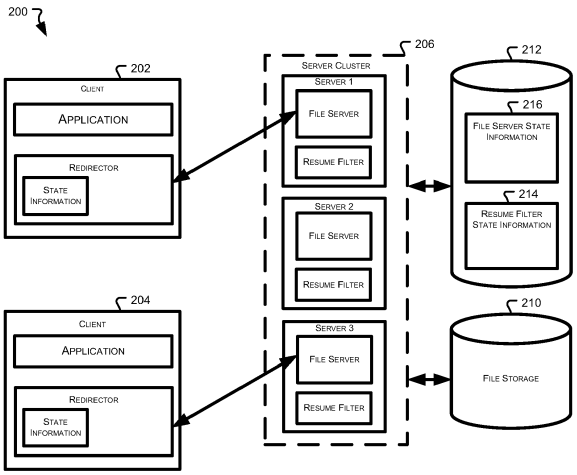

FIG. 2

EP 3 051 420 A1

Description

Background

[0001] Server clusters are commonly used to provide failover and high availability of information to clients. The use of a server cluster allows for transparent failover to clients so that any server failure is transparent to applications requesting server operations on clients. Server clusters can be useful in shared file systems to provide access to file information to several clients in a network. However, issues may arise when the shared file system utilizes a stateful protocol, such as the Server Message Block (SMB) protocol. When a server in a server cluster fails, some stateful protocols do not provide a way to transfer client state from the failed server to an alternative server. Also, file access protocols that do provide for storing some state information do not provide for different components to store different state information.

[0002] It is with respect to these and other considerations that embodiments have been made. Also, although relatively specific problems have been discussed, it should be understood that the embodiments should not be limited to solving the specific problems identified in the background.

Summary

[0003] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description section. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0004] Described are embodiments that utilize persistent handles in a shared file system. The persistent handles are used to retain state across network failures and server failovers. Persistent handles are requested by a client after a session has been established with a file server. The request for the persistent handle includes a handle identifier generated by the client. The server uses the handle identifier to associate with state information. When there is a network failure or a server failover, and a reconnection to the client, the handle identifier is used to identify replayed requests that if replayed would create an inconsistent state on the server. The replayed requests are then appropriately handled.

[0005] Embodiments may be implemented as a computer process, a computing system or as an article of manufacture such as a computer program product or computer readable media. The computer program product may be a computer storage media readable by a computer system and encoding a computer program of instructions for executing a computer process. The computer program product may also be a propagated signal on a carrier readable by a computing system and encoding a computer program of instructions for executing a computer process.

Brief Description of the Drawings

[0006] Non-limiting and non-exhaustive embodiments are described with reference to the following figures.

FIG. 1 illustrates a system that may be used to implement embodiments.

FIG. 2 illustrates a block diagram of a client and file server cluster communicating using a file access protocol consistent with some embodiments.

FIG. 3 illustrates an operational flow for providing replay defense on server failover consistent with some embodiments.

FIG. 4 illustrates operational flows for maintaining consistent availability of file information consistent with some embodiments.

FIG. 5 illustrates a block diagram of a computing environment suitable for implementing embodiments.

Detailed Description

[0007] Various embodiments are described more fully below with reference to the accompanying drawings, which form a part hereof, and which show specific exemplary embodiments. However, embodiments may be implemented in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the embodiments to those skilled in the art. Embodiments may be practiced as methods, systems or devices. Accordingly, embodiments may take the form of a hardware implementation, an entirely software implementation or an implementation combining software and hardware aspects. The following detailed description is, therefore, not to be taken in a limiting sense.

[0008] FIG. 1 illustrates a system 100 that may be used to implement some embodiments. System 100 includes clients 102 and 104 and a server cluster 106. Clients 102 and 104 communicate with server cluster 106 through network 108. Server cluster 106 stores information that is accessed by applications on clients 102 and 104. Clients 102 and 104 establish sessions with cluster 106 to access the information on cluster 106. Although in FIG. 1 only clients 102 and 104 are shown as communicating with cluster 106, in other embodiments there may be more than two clients accessing information from server cluster 106.

[0009] As shown in FIG. 1, server cluster 106 includes servers 106A, 106B, and 106C, which provide both high availability and redundancy for the information stored on cluster 106. In embodiments, the cluster 106 has a file system that is accessed by the clients 102 and 104. Although three servers are shown in FIG. 1, in other embodiments cluster 106 may include more than three serv-

ers, or fewer than three servers. In embodiments, applications on clients 102 and 104 request file information from a file system, and, transparent to the application, the file information is retrieved from a shared file system on server cluster 106.

[0010] In accordance with one embodiment, servers 106A, 106B, and 106C are utilized to provide consistent availability of the file system stored on cluster 106. This is done by utilizing components on clients 102 and 104 and servers 106A, 106B, and 106C to store state information that can be used to reestablish sessions between clients 102 and 104 and cluster 106 should there be a failure of network 108 or a failure of one of servers 106A, 106B, and 106C. As described in greater detail below, the storing of state information allows clients 102 and 104 to have consistent file access and failover that is transparent to applications running on clients 102 and 104.

[0011] The servers, e.g., 106A, 106B, and 106C, of cluster 106, in embodiments, each provide access to file information to clients and are configured to provide consistent availability of the file information to the clients. To illustrate one embodiment, client 102 may send a request to establish a session with a server of cluster 106. For example, client 102 may establish a session with server 106A to access a shared file system stored on server cluster 106. As part of the process of establishing the session, client 102 may utilize a file access protocol. In embodiments, the file access protocol is a version of the Network File System (NFS), or the Server Message Block (SMB) protocol.

[0012] The establishment of a session may involve the exchange of a number of negotiate requests and responses transmitted between client 102 and server 106A. In versions of the SMB protocol, there are specifically defined negotiate packets that are used to negotiate the exact version of the protocol that will be used during the session, as well as advertise the capabilities of both the client, e.g., 102, and server, e.g., 106A, to each other. In one embodiment, the negotiate packets may include an indication that the server 106A is part of a cluster, e.g. cluster 106. This allows the client to know that the server 106A can provide consistent availability, in other words, transparent failover capabilities.

[0013] Continuing with the example above, after the session is established, client 102 can send a message formatted according to the file access protocol to server 106A for a persistent handle to access a file in the file system. Requesting a persistent handle, in embodiments, indicates that the client would like to utilize the transparent failover capabilities available as a result of server 106A being part of cluster 106. In embodiments, the request includes a handle identifier that is a globally unique identifier.

[0014] The server 106A will receive the request for a persistent handle and store the handle identifier with state information for the session with client 102. The storing of state information may merely involve the file server persisting the handle identifier to storage and storing state information in association with the handle identifier. As described in greater detail below, in some embodiments, different types of state information may be stored using separate components, such as a filter. In yet other embodiments, information relating to persistent handles is replicated between nodes and is not stored to persistent storage on the file system. In still other embodiments, information concerning persistent handles is both replicated between nodes and is stored to persistent storage on the file system.

[0015] The server 106A sends a response to client 102 granting the persistent handle and access to file information. Client 102 can then proceed to send other requests for performing various operations on the file. For example, client 102 may send requests to read file information, write to the file, enumerate attributes of the file, close the file, and request various locks on the file. Each of the operations requested by the client may result in updating the state information to ensure that if the client is disconnected, the state of the client can be reinstated. This updating may involve saving the additional state information in association with the handle identifier.

[0016] At some point, the client 102 may be disconnected from the server. The disconnection may be because of network failure or disruptions, for example. Alternatively, the disconnection may be because of failure of server 106A. In those embodiments involving a network failure, client 102 may detect that a disconnection has occurred and wait for the network to become available to reconnect with the server 106A. In other embodiments, once client 102 detects a failure it sends a request to reconnect to cluster 106, which will provide a failover server to handle the reconnection request.

[0017] In either case, client 102 sends a request to reconnect. The request will include the handle identifier. The server 106A, or an alternative server (106B or 106C) will retrieve the state information based on the handle identifier, reestablish the previous state using the state information, and send the client a response indicating that the reconnection is successful. In some embodiments, the reconnection may not be possible, if the previous state information has been lost or is otherwise unavailable. In these situations, the server may treat the reconnection request as a request to establish a session and respond accordingly.

[0018] After the session is reestablished, client 102 sends new file access requests. In some embodiments, one of the new file access requests may be replays of previous requests. The replayed request may be of a type that if processed by the server, without recognizing that it is a replay, would create an inconsistent state on the server. The exact type of request depends upon how requests are handled by the file access protocol being used. For example, in versions of the SMB protocol, byte range locks may be requested and granted on portions of a file. Therefore, if the client sent a request to lock portions of a file and the request is completed but the

client is not notified prior to the disconnection, the client could replay the previous request. The server would need to be able to identify that the request is a replay. Therefore, in embodiments, the handle identifier sent with the original request for the persistent handle is used to identify replayed requests. Once identified, the replayed requests may be processed in order to avoid an inconsistent state on the server.

[0019] In some embodiments, in order to provide transparent failover to applications on the client 102, there may be state information that is stored on the client 102. That is, the server 106A (or a failover server) may not be responsible for storing all of the information that is necessary to restore state after a reconnection. In some embodiments, the client may be responsible for reestablishing some state. For example, if requests to read file information were sent before the disconnection, the server may not be responsible for saving state information regarding the read requests. When the reconnection occurs, the client may be responsible for resending the read requests. Additional description of embodiments, in which state information is restored by different components, is described in greater detail below with respect to FIG. 2.

[0020] The foregoing description is merely one example of how the embodiment shown in FIG. 1 may operate. As described in greater detail below, embodiments may involve different steps or operations. These may be implemented using any appropriate software or hardware component or module.

[0021] Turning now to FIG. 2, it shows a block diagram of a software environment 200 with client 202, client 204, and a server cluster 206 with three servers (server 1, server 2, and server 3). Also shown is file storage 210 where the file system stores file information and storage 212 where state information may be stored by one or more of server 1, server 2, and server 3.

[0022] As is shown in FIG. 2, client 202 and client 204 each include an application which may request file information. The application may be for example a word processing application, a spreadsheet application, a browser application or any other application which requests access to files. In the embodiment shown in FIG. 2, the files are located in a shared file system stored within file storage 210. Client 202 and client 204 each further include a redirector which redirects request for files from the applications to a file server, which provides access to the shared file system. The redirectors communicate with file servers using a file access protocol. In some embodiments, the file access protocol may be a version of NFS or of the SMB protocol. For purposes of illustration, FIG. 2 will be described assuming that the redirectors in client 202 and client 204 communicate with file servers using a version of the SMB protocol, such as SMB 2.0. Embodiments are however not limited to the use of an SMB protocol.

[0023] Server 1, server 2, and server 3 are shown in FIG. 2 as each including a file server. As noted above, the file servers may use a version of the SMB protocol to communicate with the redirectors on client 202 and client 204. Each of server 1, server 2, and server 3 also include a resume filter that is used in some embodiments to store state information for sessions established between a client redirector and a file server.

[0024] The use of the SMB protocol to establish a session between a client and a server begins with a redirector, such as the redirector on client 202, sending a negotiate request to a file server such as server 1 in server cluster 206. The redirector and file server exchange negotiate packets to negotiate the version of SMB that will be used for the session. Additionally, during the negotiation, capabilities may also be exchanged. In one embodiment, a file server may include a capability flag in a negotiate response packet sent from the file server to the client to indicate to the client that the file server supports the use of persistent handles. In some embodiments, this is done in situations in which the file server is part of a cluster that can provide consistent availability to a client by failing over to another server in the cluster. In other embodiments, stand-alone servers may also have this capability in order to be able to reconnect to clients if there is a network failure.

[0025] Once the negotiation is completed, the redirector on the client and the file server establish a session. The client redirector can then send file access requests to the file server. In one embodiment, the redirector requests a persistent handle. Versions of the SMB protocol provide for durable handles which can be used for reconnecting to clients that are disconnected. However, they do not necessarily provide for storing and reestablishing state after a client reconnects. Thus, in embodiments, the redirector can send a request for a durable handle with some additional flag and/or indicator to note that the client redirector is requesting a persistent handle. In addition, the client may include a handle identifier that can be used to identify replayed requests after reconnection. Below is one embodiment of a durable handle request structure that may be used in a version of the SMB protocol for requesting the persistent handle:

```
struct SMB2_DURABLE_HANDLE_REQUEST_V2 {
        ULONG Flags;
      GUID Handleld; // client supplied
unique ID for this handle.
        // (used to detect replays.)
    ULONG Timeout; // timeout in seconds.
  ULONG Reserved; // must be set to ZERO.}.
```

[0026] In response to the request, the file server on server 1, in embodiments, responds by granting the persistent handle and providing a file identifier to the client redirector on client 202. The client redirector is then able to access information from the file associated with the persistent handle and the file identifier. In some embodiments, the client redirector may request a persistent handle for a directory. That is, instead of the persistent handle

being associated with an individual file, the handle may be associated with a directory.

[0027] In addition to the file server on server 1 granting the persistent handle, the file server will also store state information in storage 212. The state information may be stored in association with the handle identifier generated by the client redirector and may also be stored in association with the file identifier provided to the client redirector on client 202. As described in greater detail below, the file server may directly store state information as file server state information 216. In other embodiments, the file server may utilize a resume filter to store state information. In yet other embodiments, the file server may both directly store state information and also use the resume filter for storing other state information.

[0028] After the negotiation is complete, the client redirector sends file access requests using, for example, a version of the SMB protocol. In some embodiments, the file server will store state information for each of the requests received from the client redirector. At some point in time, there may be a disconnect between client 202 and server 1, as a result of a network failure or a failure of server 1, for example. Client 202 can reestablish a connection with server 1 if the failure was based on a network failure, or with a failover server (one of server 2 or server 3). As part of the reconnection, client 202 can send a reconnect request that includes the previously provided handle identifier as well as the file identifier provided by the file server when negotiating the original session. Because the state information is available in storage 212 which is accessible by all of the servers in server cluster 206, a failover server can identify previous state information based on the handle identifier and/or the file identifier provided by the client in the reconnect request. In those embodiments where the client is attempting to reestablish a connection with server 1, the file server on server 1 can also access the state information on storage 212 to reestablish the previous state of the session with the client.

[0029] As noted above, in some embodiments, different components in environment 200 are responsible for storing different types of state information in order to provide reestablishment of state to clients that are disconnected. As shown in FIG. 2, each of the file servers includes a resume filter. The resume filter is used in embodiments to store state information for reestablishing state when a client is reconnected. The resume filter is not dependent upon the particular file access protocol used by the file server. In embodiments, the file server will first register with the resume filter in order to store particular state information. Once registered, the file server can pass state information to the resume filter, which stores the state information as resume filter state information 214 in storage 212. In addition to resume filter state information 214, the server can store separate state information, shown as file server state information 216, in storage 212. In embodiments, the different state information can be stored in a different storage location than the resume filter state information 214. The file server state information 216 and the resume filter state information 214 may be stored in any suitable way, such as log files. As described in greater detail below, the types of state information that are stored by the resume filter is, in embodiments, general information, while the server information is more specific state information.

[0030] In some embodiments, the client is also responsible for storing some state information. As shown in FIG. 2, clients 202 and 204 store state information that is used to reestablish state when a client is reconnected after a disconnect. In these embodiments, there may be some cost savings in having clients reestablish state instead of requiring the file server to store all of the state information to reestablish the state of a client when it is reconnected after a disconnect. For example, if the file server is required to store all state information, then each time there is some request received from a client redirector, with some operation to perform on a file, the file server will be required to store some information about the requests or operations. Requiring that the client redirector store some of the state information reduces the costs of a file server having to store state information for every request or operation received from the client.

[0031] As can be appreciated, the state information that is stored on different components in environment 200 depends upon different design considerations. For example, there may be some information that is important enough that requires the file server to guarantee that the state information is coherent and consistently available, in which case the information should be stored by the file server and/or the resume filter. For example, in order for a server to enforce sharing modes and ensure that new clients requesting access do not interfere with existing client's access, state information must be stored on the server, according to embodiments. Other state information may not be as critical, and some incoherency may be tolerated in the information. As an example, a client may have locally cached file properties. The cached file properties may be requested anew after a client reconnects to a file server following a disconnect.

[0032] In one embodiment, where a version of the SMB protocol is used for communication between the client redirector and the file server, the SMB protocol may provide for specific states to be stored by the various components shown in environment 200. In one embodiment, the operations available using the SMB protocol are divided into three groups. State information associated with each group is stored by different components.

[0033] The first group may be referred to generally as non-idempotent operations, meaning that if these operations are replayed, e.g., reapplied on a file after already being applied once before a client disconnect, would create an inconsistent state on the file server. In versions of the SMB protocol, byte range locks are an example of operations that require replay detection because these locks are stacked and unstacked. Other examples include appending writes and opens/creates, which can

modify disk state, for example by creating new files or overwriting existing files. In embodiments, state associated with these types of operations is stored by the file server because the file server must recognize that these operations are being replayed. In the embodiment shown in FIG. 2, state associated with these operations would be stored by the file servers that are on each of server 1, server 2, and server 3 in storage 212 as part of file server state information 216. The handle identifier provided by the client during negotiation of a session, as described above, is used in some embodiments to identify that the request is a replay of a previous request.

[0034] A second group of operations relates to data open operations. These operations may be requests to read, write, execute, or delete information in a file. In order to be able to enforce sharing modes and prevent other clients from affecting existing clients, state regarding these open operations has to be stored on the server side, according to embodiments. State regarding open operations is also stored on the server side to block local operations from interfering with persistent handles. For example, programs running on cluster nodes are prevented from modifying, or otherwise affecting, handles being reserved for clients. In embodiments, state regarding these types of operations is stored by the resume filter. As noted above, the resume filter in embodiments is not specific to the SMB protocol but can also be used when a file server is using a different file access protocol such as NFS. In the embodiment shown in FIG. 2, the resume filter on each of server 1, server 2, and server 3 stores the state information for the open operations in storage 212 as part of resume filter state information 214.

[0035] The third group of operations includes operations that if reapplied at the server would not change the final state of the server. These may be referred to as idempotent operations. Some operations in this group include but are not limited to reads, non-appending writes, deletes, renames, metadata-set operations, and metadata-query operations. Lease state also can be stored by the client and need not be persisted by the server. In embodiments, a lease is a mechanism that is designed to allow clients to dynamically alter their buffering strategy in a consistent manner in order to increase performance and reduce network use. The network performance for remote file operations may be increased if a client can locally buffer file data, which reduces or eliminates the need to send and receive network packets. A client may not have to write information into a file on a remote server if the client confirms that no other client is accessing the data. Likewise, the client may buffer read-ahead data from the remote file if the client confirms that no other client is writing data to the remote file.

[0036] According to embodiments, lease state does not need to be persisted on the server because the resume filter blocks all creates to a given file while clients are resuming their handles after a failover. This implicitly provides a guarantee that handle leases will never be lost during the failover process if clients reconnect/resume their handles during the grace period. In other words, clients will always get back their handle leases during the resume phase. Furthermore, exclusive leases such as read/write, read/write/handle leases are granted to only a single client at any given time. This implies that there are no other data opens to the file from any other client. So during failover, since the resume filter will not allow new creates to the file until the client holding the exclusive lease has resumed all its handles, there is a guarantee that the client will get back its exclusive lease. Shared leases which do not require an acknowledgement, such as read lease, can be lost at any time without the knowledge of either server or the resume filter because the underlying file system allows the operation which caused the break to proceed. For such leases, the client, in embodiments, assumes that the lease is broken across a failover and purges its cache to prevent stale reads. State for the operations in the third group can therefore be recreated by the client without any additional support from the server. In the embodiment shown in FIG. 2, the redirectors on clients 202 and 204 store the state information for the third group of operations.

[0037] In operation, environment 200 allows applications on clients 202 and 204 to request access to files that are stored in file storage 210 in a shared file system. The applications can transparently request file information. The redirectors on the clients will establish a session with one of the servers in cluster 206, as described above, requesting a persistent handle so that the redirector can reconnect and reestablish the session should there be a disconnect. The file server will store state information in storage 212 either directly as file server state information 216 or as resume filter state information 214 using a resume filter. In some embodiments, the client will also store some state information. In the event of a disconnect, the redirector can request to reconnect to the file server, or to a failover server. The state information stored on the server side, e.g., in storage 212, and the client side can then be used to reestablish the previous state of the client. This all occurs transparent to the applications on clients 202 and 204.

[0038] As may be appreciated, the above description of environment 200 is not intended to limit the embodiments described herein. FIG. 2 and its description are merely intended to illustrate implementation of some embodiments. In other embodiments, different types of state information may be stored on different components in environment 200. Also, as indicated above, different file access protocols may be used which may determine the type of state information stored as well as what component stores the state information. Thus, embodiments are not limited to what is shown and described in FIG. 2.

[0039] The description of FIGS. 3 and 4 below is made using the server message block (SMB) protocol as the file access protocol. However, embodiments are not limited thereto. Any file access protocol including different versions of SMB or the network file system (NFS) may be used in embodiments as the file access protocol. SMB

is being used in the description merely for convenience and ease of illustration.

[0040] FIGS. 3 and 4 illustrate operational flows 300 and 400 according to embodiments. Operational flows 300 and 400 may be performed in any suitable computing environment. For example, the operational flows may be executed by systems and environments such as illustrated in FIGS. 1 and 2. Therefore, the description of operational flows 300 and 400 may refer to at least one of the components of FIGS. 1 and 2. However, any such reference to components of FIGS. 1 and 2 is for descriptive purposes only, and it is to be understood that the implementations of FIGS. 1 and 2 are non-limiting environments for operational flows 300 and 400.

[0041] Furthermore, although operational flows 300 and 400 are illustrated and described sequentially in a particular order, in other embodiments, the operations may be performed in different orders, multiple times, and/or in parallel. Further, one or more operations may be omitted or combined in some embodiments.

[0042] Operational flow 300 illustrates steps for providing replay defense on server failover. In embodiments, flow 300 illustrated in FIG. 3 may be performed by a file server that is running on a server that is part of a server cluster, e.g., server 1, server 2, and server 3 of cluster 206 (FIG. 2). Flow 300 begins at operation 302 where a request to connect to a file server is received. The request received at operation 302 is a request to establish a session with the file server in order to access file information stored on a shared file system accessible through the file server. The request may be sent by a client, e.g., clients 202 and 204 (FIG. 2). After operation 302, flow 300 passes to operation 304 where a response is sent indicating that a session has been established. In some embodiments, the request and response sent at operations 302 and 304 may be part of a number of messages that are exchanged between a client and a server to negotiate a session. The exchange of messages may include an exchange of capabilities including the capability of the file server to provide persistent handles.

[0043] Operational flow passes from operation 304 to operation 306 where a second request is received for a persistent handle. The request is sent by the client and includes a handle identifier that is generated by the client. The handle identifier is used in embodiments by the server to store state information regarding the session established between the client and the file server. As part of storing the state information, flow 300 may include, in embodiments, operation 308 in which the file server registers with a resume filter in order to store some state information. In embodiments, the resume filter is located between the protocol layer and the underlying storage system and can be used in embodiments to store state information regarding a session established between the file server and the client.

[0044] At operation 310 the state information is stored in association with the handle identifier. The state information may be stored in any appropriate form, such as in a table, database, or log file. The storage is persistent and available to the file server for reestablishing state when necessary. The state information may be stored directly by the file server. In other embodiments, flow 300 includes operation 312, in which the resume filter is used to store state information. As indicated above, the file server may register with the resume filter in some embodiments to store state information.

[0045] Flow 300 passes from operation 312 to operation 314 where a response is sent to the client granting access to the file using the persistent handle. The response includes a file identifier that is provided by the file server in the response and is also stored in association with the state information stored at operation 310, and optionally at operation 312.

[0046] Flow 300 then passes to operation 316, where optionally a number of file access requests are received. The file access requests may include a number of file operations to perform on the file associated with the persistent handle. The operations may be, for example, opens to read/write data, enumerate attributes, lease requests to allow caching of data locally, or other file access operations. The various states associated with receiving the file access requests at operation 316 may be updated at operation 318. That is, when these requests are granted to the client, the state information stored in the previous operations (310 and 312) is updated to reflect the additional state information.

[0047] After operation 318, there are a number of additional operations identified within box 319. These operations may be performed as a result of the client being disconnected from the file server. As can be appreciated, in those situations where the file server that originally performed operations 302-318 is unavailable because of a failure, the additional operations within box 319 are performed by a failover server. In other embodiments, where the failure is a result of a network problem, the operations within box 319 are performed by the same file server.

[0048] At operation 320, a request to reconnect is received. The request includes the file handle previously provided by the file server, as well as the handle identifier that the client used when requesting the persistent handle. The file server that receives the request at operation 320 can use the handle identifier and the file identifier to look up the state information. As indicated above, this operation may involve using the resume filter in order to retrieve the state information that was previously saved using the resume filter.

[0049] Flow 300 passes from operation 320 to operation 322 where the state information is used to reestablish the connection and previous state with the client. After operation 322, flow passes to operation 324 where new file access requests are received. Operation 324 therefore may include a number of operations that each includes receiving a file access request from the client.

[0050] Some of the requests received at operation 324 may be replays of previous requests that were sent prior

to the disconnect between the file server and the client. As a result, some of these operations if reapplied at the file server may create an inconsistent state. At operation 326, the new file access requests that are replays are detected. In embodiments, this operation may involve identifying the file access requests using the handle identifier previously provided by the client. Once the replay is detected at operation 326, the requests are properly processed at operation 328. That is, if the replayed operations would create an inconsistent state on the file server, they may be ignored if the previous operation was successfully performed. Alternatively, if the previous operation was not successfully performed, then the replayed operation may be applied. Flow 300 then ends at 330.

[0051] Operational flow 400 illustrates steps for maintaining consistent availability. In embodiments, flow 400 may be performed by redirectors on clients, such as clients 202 and 204 (FIG. 2), that are communicating with a file server to access files in a shared file system. The client communicates, in embodiments, with the file server using a file access protocol such as a version of the SMB protocol or a version of NFS.

[0052] Flow 400 begins at operation 402 where a request to connect to the file server is sent. The request sent at operation 402 is a request to establish a session with the file server in order to access file information stored on a shared file system accessible through the file server. The request may be sent to a file server on a server, e.g., server 1, server 2, and server 3, that is part of a server cluster (FIG. 2). The request is formatted according to a file access protocol such as a version of SMB or NFS.

[0053] After operation 402, flow 400 passes to operation 404 where a response is received indicating that a session has been established. In some embodiments, operations 402 and 404 may be part of a number of messages that are exchanged between a client and a server to negotiate a session. The exchange of messages may include an exchange of capabilities including the capability of the file server to provide persistent handles.

[0054] Operational flow passes from operation 404 to operation 406 where a request is sent for a persistent handle. As a result of the negotiating process (operations 402 and 404), the client may have been notified that the file server is capable of providing persistent handles. In order to ensure that applications on the client can have their states reestablished after a disconnect and reconnection, the client may request a persistent handle at operation 406. The request includes a handle identifier that is generated by the client.

[0055] Flow 400 passes from operation 406 to operation 408 where a response is received granting access to the file using the persistent handle. The response includes a file identifier that is provided by the file server in the response.

[0056] At operation 410 state information may, in some embodiments, be stored by the client. The state information is stored in association with the handle identifier and the file identifier provided in the response received granting the persistent handle. The state information may be stored in any appropriate form, such as in a table, database, or log file. The storage is persistent and available to the client for reestablishing state when necessary. As can be appreciated, the state information stored by the client is, in embodiments, state information for operations that can be safely replayed back to the file server without creating an inconsistent state on the file server. The replayed operations may be, for example, leases for locally caching data, reads, writes, deletes, and meta-data enumerations.

[0057] Flow 400 passes from operation 410 to operation 412 where the client sends a number of file access requests. Operation 412 may thus involve the sending of several requests to perform file operations, according to embodiments. Following operation 412 is operation 414, where state information on the client is updated. As may be appreciated, operations 414 may occur numerous times, namely each time that a file access request is sent by the client at operation 412.

[0058] From operation 414, flow passes to operation 416 where a disconnect is detected. The detection may occur by virtue of a timeout, an event notification or some other means. Following operation 416, a request is sent to reconnect and reestablish the session previously established with the file server at operation 418. The request includes the file handle previously provided by the file server, as well as the handle identifier that the client used when requesting the persistent handle.

[0059] Flow 400 passes from operation 418 to operation 420 where a determination is made that the reconnect is successful. After operation 420, flow passes to operation 422 where state information stored on the client is used to reestablish the previous state. Operation 422 may involve sending a number of different requests, including read, write, enumerate, requests for locks or other operations to reestablish the previous state. Flow passes from operation 422 to operation 424, where the client sends new file access requests. Flow ends at 426.

[0060] FIG. 5 illustrates a general computer system 500, which can be used to implement the embodiments described herein. The computer system 500 is only one example of a computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the computer and network architectures. Neither should the computer system 500 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the example computer system 500. In embodiments, system 500 may be used as a client and/or server described above with respect to FIG. 1.

[0061] In its most basic configuration, system 500 typically includes at least one processing unit 502 and memory 504. Depending on the exact configuration and type of computing device, memory 504 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.)

or some combination of the two. This most basic configuration is illustrated in FIG. 5 by dashed line 506. In embodiments, system memory 504 stores applications such as application 523, which requests access to file information. System memory 504 also includes redirector 522 that intercepts the requests and communicates them to a file server, according to embodiments.

[0062] The term computer readable media as used herein may include computer storage media. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory 504, removable storage, and non-removable storage 508 are all computer storage media examples (i.e., memory storage). In embodiments, data, such as state information 520, for example, are stored. Computer storage media may include, but is not limited to, RAM, ROM, electrically erasable read-only memory (EEPROM), flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store information and which can be accessed by computing device 500. Any such computer storage media may be part of device 500. Computing device 500 may also have input device(s) 514 such as a keyboard, a mouse, a pen, a sound input device, a touch input device, etc. Output device(s) 516 such as a display, speakers, a printer, etc. may also be included. The aforementioned devices are examples and others may be used.

[0063] The term computer readable media as used herein may also include communication media. Communication media may be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" may describe a signal that has one or more characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), infrared, and other wireless media.

[0064] Reference has been made throughout this specification to "one embodiment" or "an embodiment," meaning that a particular described feature, structure, or characteristic is included in at least one embodiment. Thus, usage of such phrases may refer to more than just one embodiment. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

[0065] One skilled in the relevant art may recognize, however, that the embodiments may be practiced without one or more of the specific details, or with other methods, resources, materials, etc. In other instances, well known structures, resources, or operations have not been shown or described in detail merely to avoid obscuring aspects of the embodiments.

[0066] While example embodiments and applications have been illustrated and described, it is to be understood that the embodiments are not limited to the precise configuration and resources described above. Various modifications, changes, and variations apparent to those skilled in the art may be made in the arrangement, operation, and details of the methods and systems disclosed herein without departing from the scope of the claimed embodiments.

Numbered Embodiments forming part of the description

[0067]

1. A computer implemented method of providing consistent availability to clients accessing a shared file system on a server cluster, the method comprising:

    receiving at a file server a request to connect to the file server to access file information in a shared file system, the first request being formatted according to a file access protocol, wherein the file server is one of a plurality of servers in a server cluster;
    sending a response from the file server, the response establishing a session with a client for allowing access to file information in the shared file system, the response being formatted according to the file access protocol;
    receiving a request at the file server to open a persistent handle on the file server for accessing a file in the shared file system by the client, the request including a handle identifier provided by the client;
    in response to receiving the request, the file server:

        storing first state information about the session in association with the handle identifier; and
        sending a response to the client granting access to the file;

    after a client disconnect, receiving a request to reestablish the session using the persistent handle; and
    reestablishing the session using the first state information.

2. The method of embodiment 1, wherein the first state information comprises state of an operation that if resent by the client causes the file server to end up in an inconsistent state.

3. The method of embodiment 1, further comprising:

after the reestablishing the session, receiving a new request from the client, the request including the handle identifier.

4. The method of embodiment 1, wherein the client disconnect occurs because of a failure of the file server and the reestablishing the connection is performed by a second file server in the server cluster.

5. A computer readable storage medium comprising computer executable instructions that when executed by a processor perform a method of maintaining consistent state, the method comprising:

sending a request by a client to connect to a server to access file information, the request being formatted according to a file access protocol;
receiving a response from the server, the response establishing a session with the client for allowing access to file information on the server, the response being formatted according to the file access protocol;
sending a request to open a persistent handle on the server for accessing a file on the server by the client, the request including a handle identifier provided by the client;
receiving a response at the client granting access to the file;
detecting that the client has been disconnected from the server;
sending a request to reestablish the session using the persistent handle, the request to reestablish the session including the handle identifier;
determining that the session has been reestablished; and
sending a new request.

6. A system for providing consistent availability of file information, the system comprising:

a first server comprising:

at least one processor configured to execute computer executable instructions;
at least one computer readable storage media storing the computer executable instructions that when executed by the at least one processor provide:

a first file server configured to:

receive a request to open a persistent handle for accessing a file on the first file server by a client, the request including a handle identifier provided by the client;
store first state information in association with the handle identifier;
register with a resume key filter to store second state information in association with a resume key; and
send a response to the client granting access to the file;

the resume key filter configured to:

receive a registration request from the first file server;
store the second state information with the resume key; and
send the second state information to the first file server in response to a request from the first file server for the second state information.

7. The system of embodiment 6, wherein the system further comprises:

at least one client, comprising:

at least one additional processor configured to execute computer executable instructions;
at least one additional computer readable storage media storing the computer executable instructions that when executed by the at least one additional processor provide:

a file access redirector configured to:

send the request to open the persistent handle on the first server for accessing the file on the first server by the client, the request including the handle identifier;
receive the response granting access to the file;
detect that the client has been disconnected from the first file server;
send a request to reestablish a session using the persistent handle,

the request to reestablish the session including the handle identifier;

determine that the session has been reestablished; and
send a new request.

8. The system of embodiment 6, wherein the first file server is one of a plurality of servers in a server cluster, and wherein a second server of the plurality of servers in the server cluster comprises:

at least one additional processor configured to

execute computer executable instructions;
at least one additional computer readable storage media storing the computer executable instructions that when executed by the at least one additional processor provide:

a second file server configured to:

receive a request to reestablish a session using the persistent handle, the session previously established by the first file server.

9. The system of embodiment 8, wherein the second server of the plurality of servers is further configured to:

use the state information of the first file server to reestablish a previous state of the session.

10. The system of embodiment 7, wherein the file access redirector uses a version of a Server Message Block (SMB) protocol to request file operations from the first file server, and the first file server uses the version of the SMB protocol to communicate with the file access redirector.

Claims

1. A computer implemented method of providing consistent availability to clients accessing a shared file system on a server cluster, the method comprising:

receiving at a file server a request to connect to the file server to access file information in a shared file system, the first request being formatted according to a file access protocol, wherein the file server is one of a plurality of servers in a server cluster;
sending a response from the file server, the response establishing a session with a client for allowing access to file information in the shared file system, the response being formatted according to the file access protocol;
receiving a request at the file server to open a persistent handle on the file server for accessing a file in the shared file system by the client, the request including a handle identifier provided by the client;
in response to receiving the request, the file server:

storing first state information about the session in association with the handle identifier; and
sending a response to the client granting access to the file;

after a client disconnect, receiving a request to reestablish the session using the persistent handle; and
reestablishing the session using the first state information.

2. The method of claim 1, wherein the first state information comprises state of an operation that if resent by the client causes the file server to end up in an inconsistent state.

3. The method of claim 1 or claim 2, wherein byte range locks are requested and granted on portions of a file.

4. The method of any of claims 1 to 3, further comprising the file server registering with a resume filter in order to store particular state information.

5. The method of any of claims 1 to 4, further comprising after the reestablishing the session, receiving a new request from the client, the request including the handle identifier.

6. The method of any of claims 1 to 5, wherein the handle identifier sent with the original request for the persistent handle is used to identify replayed requests.

7. A system for maintaining consistent availability comprising at least one computer readable storage media storing computer readable instructions that, when executed, provide a method, the method comprising:

sending a request to connect to a file server to access file information, the request being formatted according to a file access protocol;
receiving a response, the response indicating that a session has been established;
sending a request for a persistent handle, the request including a handle identifier generated by the client;
receiving a response granting access to the file using the persistent handle, the response including a file identifier that is provided by the file server in the response;
the client storing state information, the state information being stored in association with the handle identifier and the file identifier provided in the response received granting the persistent handle, wherein the state information stored by the client is state information for operations that can be safely replayed back to the file server without creating an inconsistent state on the file server;
detecting a disconnect;
sending a request to reconnect and reestablish the session previously established with the file

server, the request including the file handle previously provided by the file server, as well as the handle identifier that the client used when requesting the persistent handle,
making a determination that the reconnect is successful;
using state information stored on the client to reestablish the previous state.

8. The system of claim 7, wherein the client disconnect occurs because of a failure of the file server and the reestablishing the connection is performed by a second file server in the server cluster.

9. The system of claim 7 or claim 8, wherein the handle identifier sent with the original request for the persistent handle is used to identify replayed requests.

10. The system of any of claims 7 to 9, wherein byte range locks are requested and granted on portions of a file.

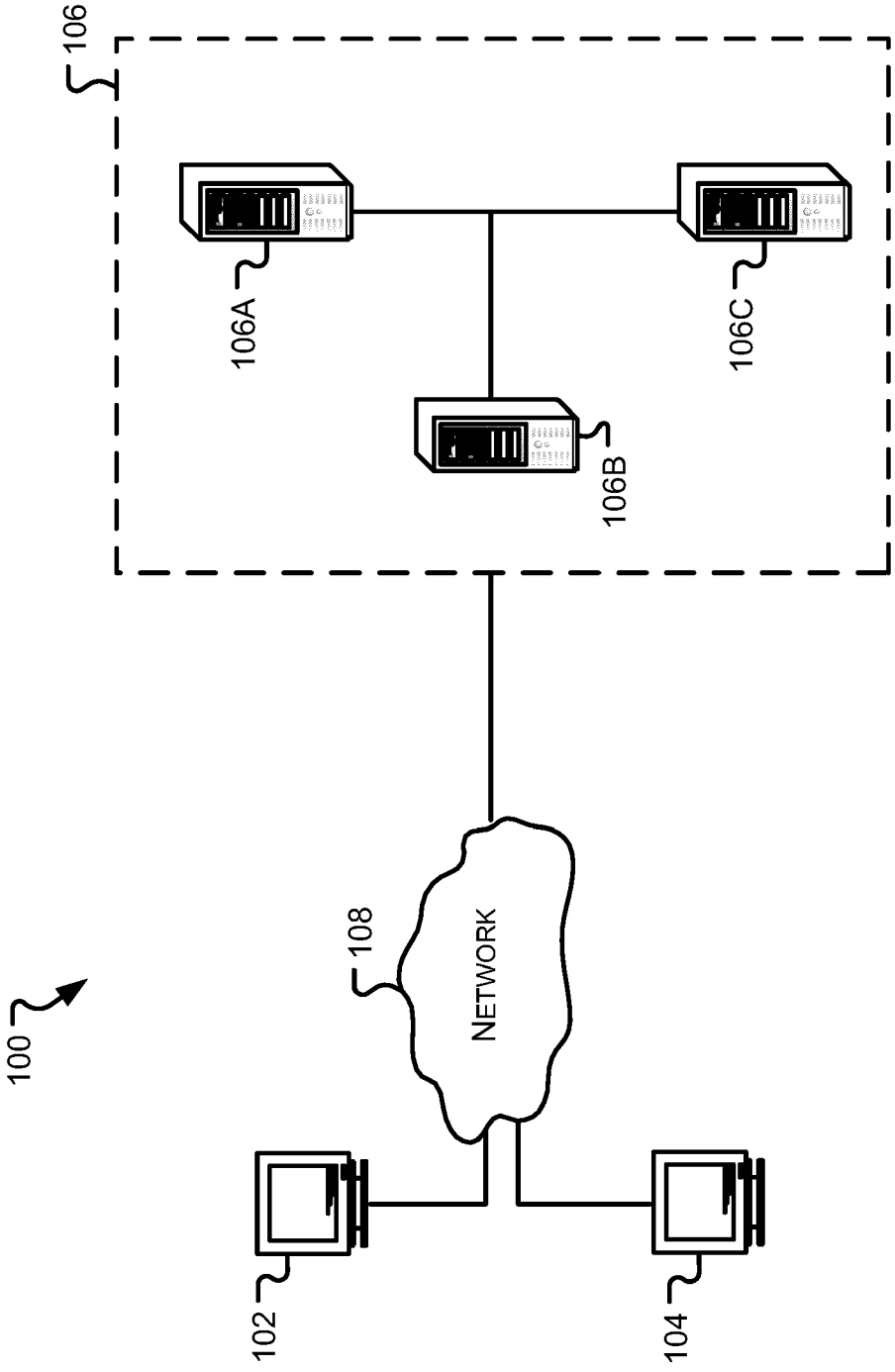

FIG.1

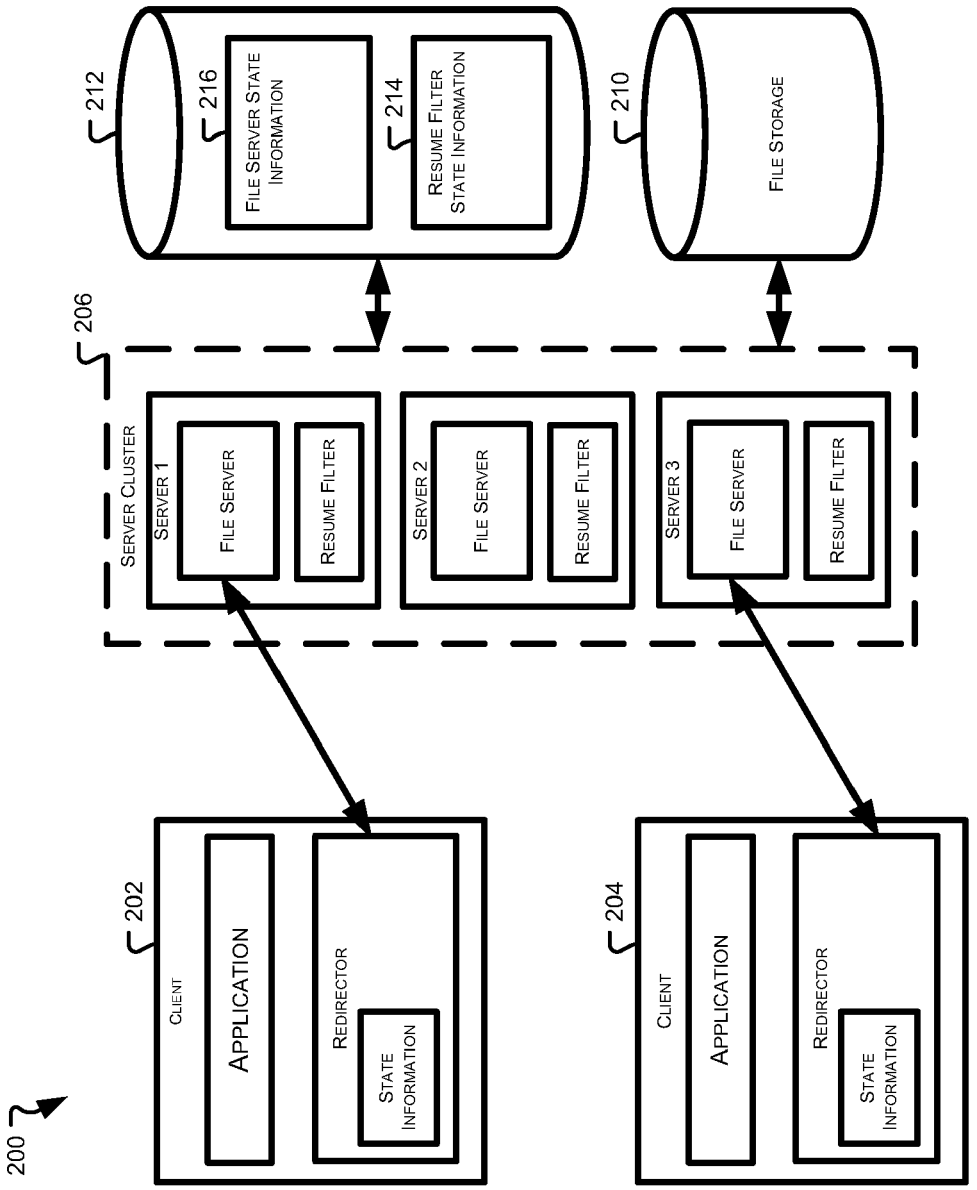
200
202
CLIENT
APPLICATION
REDIRECTOR
STATE INFORMATION
204
CLIENT
APPLICATION
REDIRECTOR
STATE INFORMATION
206
SERVER CLUSTER
SERVER 1
FILE SERVER
RESUME FILTER
SERVER 2
FILE SERVER
RESUME FILTER
SERVER 3
FILE SERVER
RESUME FILTER
210
FILE STORAGE
212
216
FILE SERVER STATE INFORMATION
214
RESUME FILTER STATE INFORMATION

FIG. 2

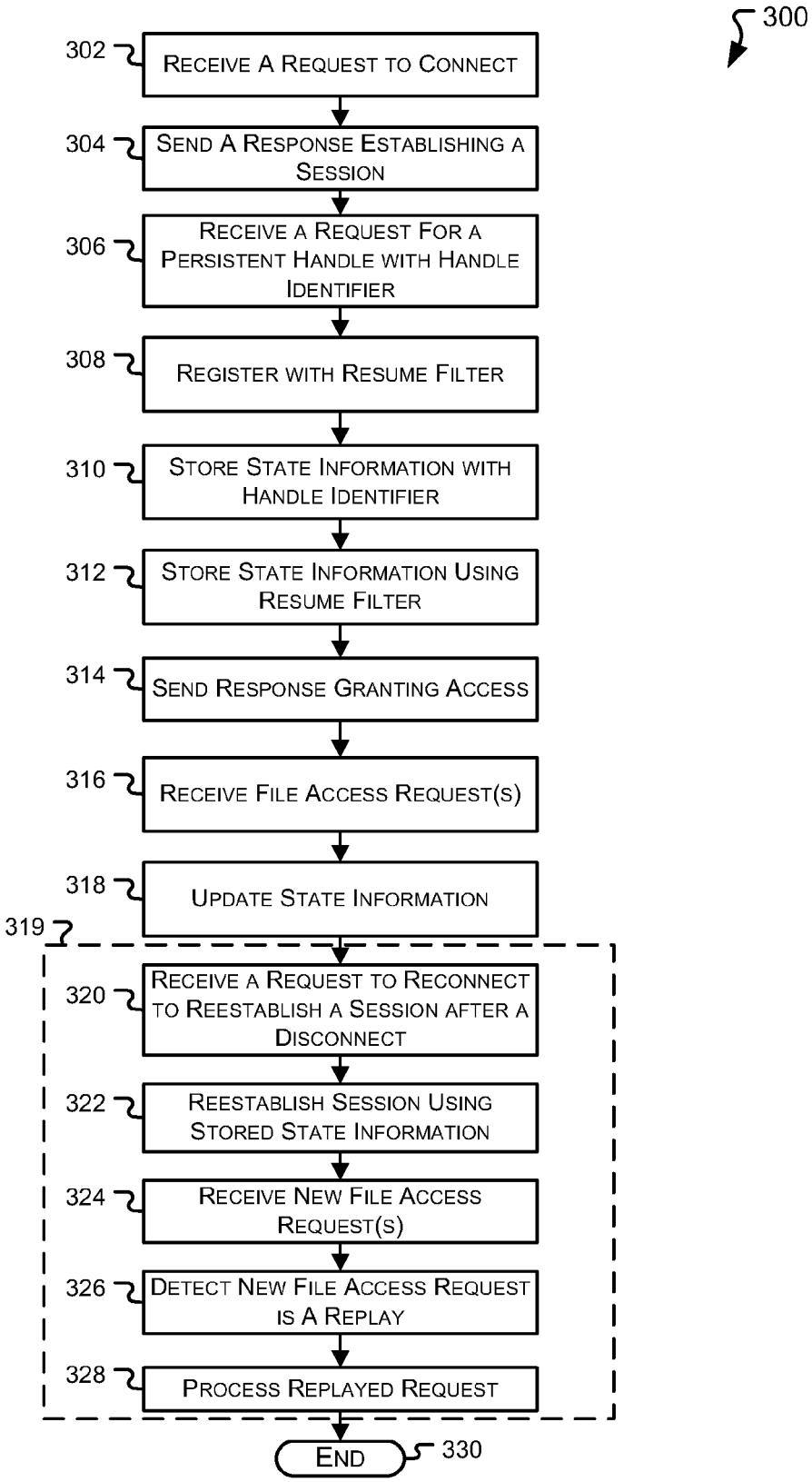
300
302
RECEIVE A REQUEST TO CONNECT
304
SEND A RESPONSE ESTABLISHING A SESSION
306
RECEIVE A REQUEST FOR A PERSISTENT HANDLE WITH HANDLE IDENTIFIER
308
REGISTER WITH RESUME FILTER
310
STORE STATE INFORMATION WITH HANDLE IDENTIFIER
312
STORE STATE INFORMATION USING RESUME FILTER
314
SEND RESPONSE GRANTING ACCESS
316
RECEIVE FILE ACCESS REQUEST(S)
318
UPDATE STATE INFORMATION
319
320
RECEIVE A REQUEST TO RECONNECT TO REESTABLISH A SESSION AFTER A DISCONNECT
322
REESTABLISH SESSION USING STORED STATE INFORMATION
324
RECEIVE NEW FILE ACCESS REQUEST(S)
326
DETECT NEW FILE ACCESS REQUEST IS A REPLAY
328
PROCESS REPLAYED REQUEST
END
330

FIG.3

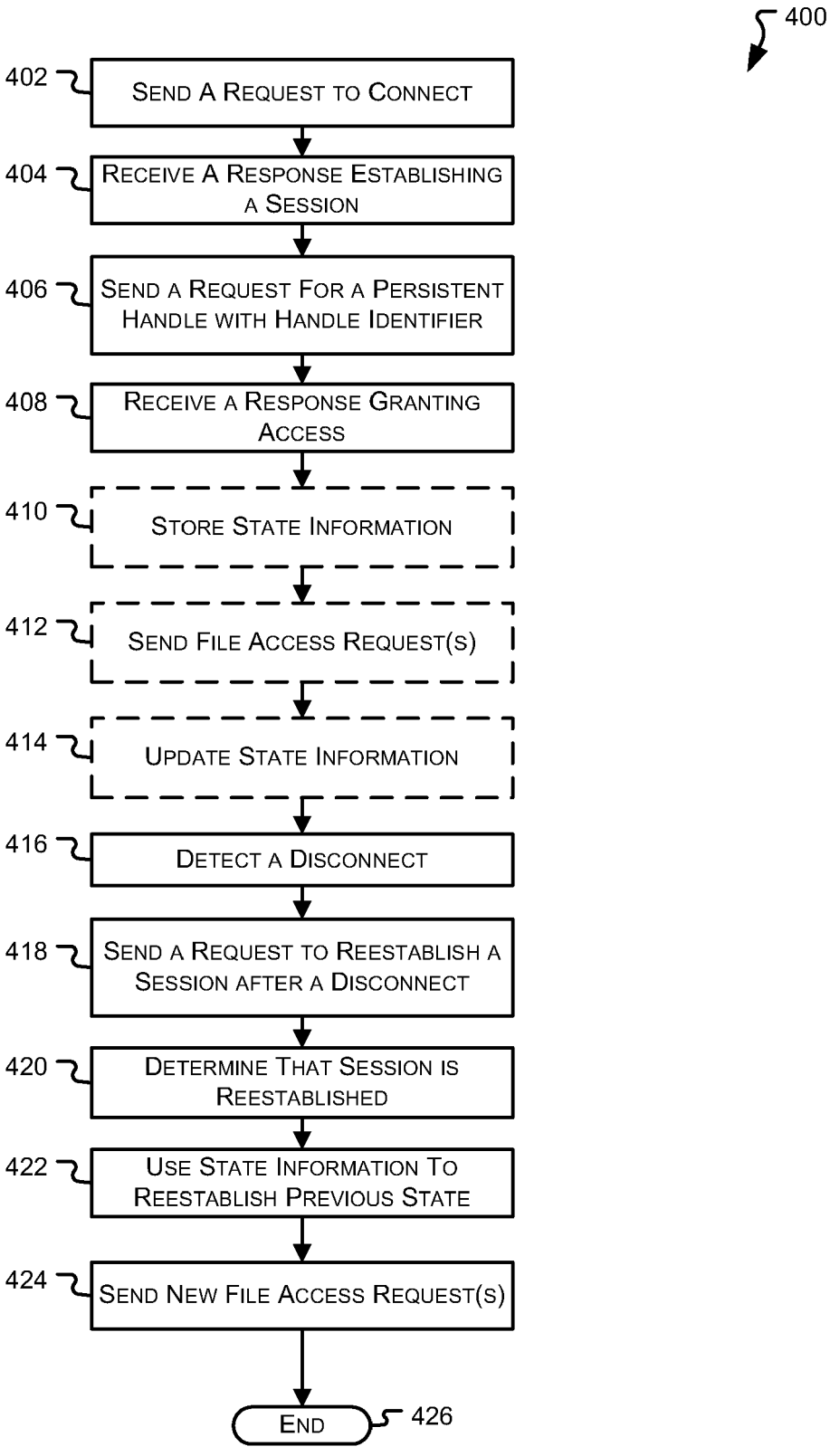
400
402
SEND A REQUEST TO CONNECT
404
RECEIVE A RESPONSE ESTABLISHING A SESSION
406
SEND A REQUEST FOR A PERSISTENT HANDLE WITH HANDLE IDENTIFIER
408
RECEIVE A RESPONSE GRANTING ACCESS
410
STORE STATE INFORMATION
412
SEND FILE ACCESS REQUEST(S)
414
UPDATE STATE INFORMATION
416
DETECT A DISCONNECT
418
SEND A REQUEST TO REESTABLISH A SESSION AFTER A DISCONNECT
420
DETERMINE THAT SESSION IS REESTABLISHED
422
USE STATE INFORMATION TO REESTABLISH PREVIOUS STATE
424
SEND NEW FILE ACCESS REQUEST(S)
END
426

FIG.4

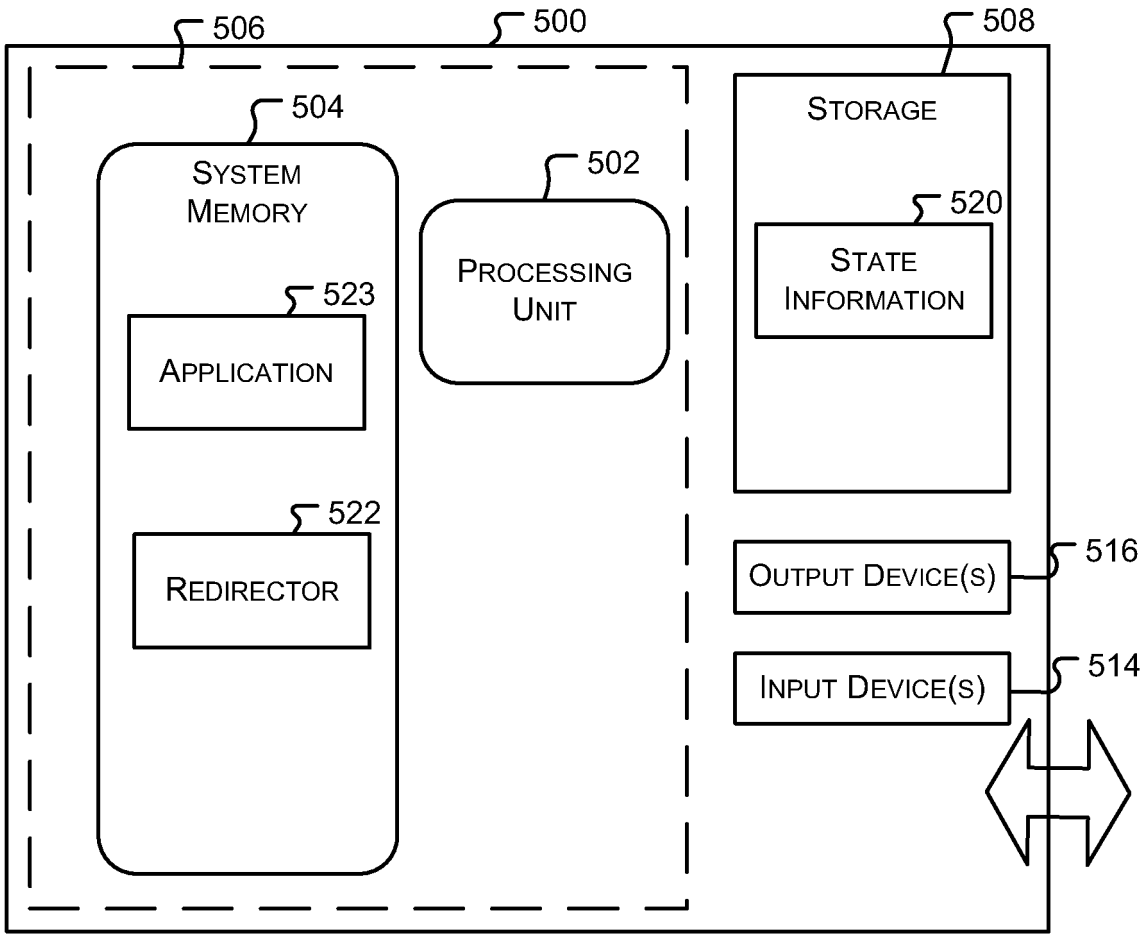
506
500
508
504
SYSTEM MEMORY
502
PROCESSING UNIT
STORAGE
520
STATE INFORMATION
523
APPLICATION
522
REDIRECTOR
OUTPUT DEVICE(S)
516
INPUT DEVICE(S)
514

FIG.5

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 16 15 9584

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
Y	US 2007/220155 A1 (NALLA AMAR [US] ET AL) 20 September 2007 (2007-09-20) * paragraphs [0028] - [0036], [0043] - [0046], [0050] - [0053] * * figures 3,4,6,7 * * claim 1 *	1-10	INV. G06F11/14 G06F11/20 H04L12/24
Y	US 2005/198380 A1 (PANASYUK ANATOLIY [US] ET AL) 8 September 2005 (2005-09-08) * paragraphs [0084] - [0087], [0091] - [0093], [0096] - [0100], [0146] - [0148] * * figures 1a,1b,4,7 * * claim 1 *	1-10	
X A	US 5 987 621 A (DUSO WAYNE W [US] ET AL) 16 November 1999 (1999-11-16) * column 2, line 31 - column 3, line 11 * * column 6, line 36 - column 8, line 16 * * column 35, line 40 - column 36, line 19 * * column 38, line 53 - column 39, line 42 * * column 45, line 9 - column 46, line 54 * * column 48, line 8 - column 50, line 35 * * column 54, line 25 - column 55, line 39 * * figures 1,2,29,30,34-37,41-43 *	1,7 2-6,8-10	TECHNICAL FIELDS SEARCHED (IPC) G06F
A	EP 1 669 850 A1 (HITACHI LTD [JP]) 14 June 2006 (2006-06-14) * paragraphs [0002], [0005] - [0010], [0012] - [0032], [0045] - [0101], [0120] - [0161] * * figures 1-3, 7-10, 11A, 11B *	1-10	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	23 June 2016	Johansson, Ulf

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 16 15 9584

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

23-06-2016

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007220155 A1	20-09-2007	US 2007220155 A1	20-09-2007
		US 2011307605 A1	15-12-2011
US 2005198380 A1	08-09-2005	AT 381196 T	15-12-2007
		AU 2004306772 A1	21-04-2005
		CA 2541151 A1	21-04-2005
		DE 602004010703 T2	27-11-2008
		EP 1678918 A1	12-07-2006
		ES 2298835 T3	16-05-2008
		JP 2007515852 A	14-06-2007
		KR 20060126952 A	11-12-2006
		US 2005198380 A1	08-09-2005
		WO 2005036858 A1	21-04-2005
US 5987621 A	16-11-1999	NONE	
EP 1669850 A1	14-06-2006	EP 1669850 A1	14-06-2006
		JP 4451293 B2	14-04-2010
		JP 2006164169 A	22-06-2006
		US 2006129513 A1	15-06-2006
		US 2009144290 A1	04-06-2009

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82